Taarini Ananya Homework 3

1) Octal unassigned: 62 and 12. Octal is first converted to decimal
62 → 6·8 + 2 = 50, 12 → 1·8 + 2 = 10
62 (50) · 12 (10) = 500 decimal then converted to binary is 50 → 110 010
10 → 001010.

Step 0 initial values	001010 × 110010	→	000000000000
Step 1	001010 × 110010	→	000000000000
	001010 × 001100100	→	000000000000
	00010 × 001100100	→	000000000000
Step 2	000101 × 001100100	→	000001100100
	000101 × 011001000	→	000 001 100 100
	000010 × 011001000	→	000 001100100
Step 3	000010 × 011001000	→	000 001100100
	000010 × 110010000	→	000 001100100
	000001 × 110010000	→	000 001100100
Step 4	000001 × 000110010000	→	000111110 100
	000001 × 001100100000	→	000111110100
	000000 × 001100100000	→	000111110 100
Step 5	000000 × 001100100000	→	000111110 100
	000000 × 011001000000	→	000111110100
	000000 × 011001000000	→	000 111110100
Step 6	000000 × 110010000000	→	000111110100
	000000 × 110010000000	→	000111110100
	000000 × 110010000000	→	000111 110100

$62_{10} \times 12_{10} = 50_{8} \times 100_{8} = 110010 \times 001010_{2} = 000111110100_{2}$

To multiply these two octal values, they are first converted to decimal using [first #] × 8 + [second #]. This resulted in 50 and 100. Then, these values are converted to binary as 110010 and 001010. Finally, the multiplicand is shifted left and multiplier shifted right for each step and multiplied. equaling 000111110100_{10}, or 500 decimal, 744 octal.

2. The best way to calculate 0x33 x 0x55 using shifts and add/subtracts while assuming both inputs are 8-bit signed integers is to first convert both hex numbers to decimal. Next, they need to be converted to binary and split up with their exponents. The multiplicand is then shifted according to the multipliers' exponents. The results are added to get the final value.

Hex to decimal: $0x33 \rightarrow (3 \times 16^1) + (3 \times 16^0) = 48 + 3 = 51$

$0x55 \rightarrow (5 \times 16^1) + (5 \times 16^0) = 85$

Decimal to binary: $51 \rightarrow 110011 \rightarrow 2^5(32) + 2^4(16) + 2^1(2) + 2^0(1)$

Shift: $(85 \times 32) + (85 \times 16) + (85 \times 2) + (85 \times 1) =$

Add: $2720 + 1360 + 170 + 85 = 4335$

Convert: $4335_{10} \rightarrow 0x10EF$ (hex)

4. In order to find if the bit pattern 0x0C000000 represents a floating point number using the IEEE 754 standards, it needs to be converted to binary first. Then, the binary needs to split for the sign, exponent and mantissa. Next, these values can be put together using the 754 formula: $(-1)^S \times F \times 2^E$.

Hex to decimal: $C \rightarrow 1100$ $0 \rightarrow 0000 = 0000$

0x0C000000 = 0000 1100 0000, 0000 0000 0000 0000 0000

IEEE 754 standard: Sign: $0 \rightarrow$ positive

Exponent: 00011000, mantissa: 0000 0000 0000 0000 0000

Exponent: $00011000 = 24 - 127 = -103$

$(-1)^0 + 1.0 \times 2^{(-103)} = 9.86 \times 10^{-32}$

Yes, after converting and calculating, 0x0C000000 is a floating point number interpreted as a 32-bit IEEE 754 standard equaling 9.86×10^{-32}

5. To write down the binary representation of the decimal number 63.25 assuming the IEEE 754 double precision format, 63.25 needs to be converted to binary, normalized, divided to the IEEE 754's fields, and represented as a whole.

Decimal to binary: $63_{10} \rightarrow 111111_2$ and $.25_{10} \rightarrow 0.01_2 = 111111.01_2$

Normalize: $1.xxxxxx \cdot 2^c \rightarrow 1.1111101 \times 2^5$

Develop fields: Mantissa = 1111101 Exponent: 5 Sign: 0 (positive)

$5 + 127 = 132 \rightarrow 10000100$ exponent

IEEE 754 double standard:

$0 \; \underbrace{00000000000000010000100}_{23 \text{ bits}} \; \underbrace{000\ldots1111101}_{52 \text{ bits}}$

3. To calculate the octal values of 74 by 21, they first need to be converted to decimal, then binary. Next, the values need to be shifted/calculated according to the table

Octal to decimal → $7 \cdot 8 + 4 = 60$ $2 \cdot 8 + 1 = 17$

Decimal to binary → $60 = 111100$ $17 = 10001$

Step	Quotient	Divisor		Remainder
Step 0	000 000	010 001000 000	→	000000111 100
1	000 000	010 001000000	→	101111111100
	000 000	010 001000000	→	000000111100
	000000	001 000 100 000	→	000000111100
2	000000	001000100 000	→	111 000 011100
	000000	001000100000	→	000 000 111 100
	000000	000 100010 000	→	000 000 111 100
3	000000	000 100 010000	→	111100 101 100
	000000	000 100 010000	→	000 000 111 100
	000000	000 010 001000	→	000000111100
4	000000	000 010001 000	→	111 110 110 100
	000000	000 010001000	→	000 000 111 100
	000000	000 001000 100	→	000 000 111 100

5	000000	000001000100	→	111111111000
	000000	000001000100	→	000000111100
	000000	000000100010	→	000000111100
6	000000	000000100010	→	000000011010
	000001	000000100010	→	000000011010
	000001	000000010001	→	000000011010
7	000001	000000010001	→	000000001001
	000011	000000010001	→	000000001001
	000011	00000000100	→	000000001001

The binary numbers according to the provided table at each step are rem = rem - div, rem < 0 → + Div, SLL Q, Q0 = 0 or rem ≥ 0 → SLL Q, Q0 = 1, and Shift div right. This is repeated for seven steps until all numbers are exhausted. The octal numbers 74 and 21 as decimal 60 and 17, binary as 111100 and 10001 are divided to equal a quotient of 000011, divisor of 0000000100, and remainder of 000000001001. Decimal 60/17 is 3 remainder 9 and 74/21 is octal 3_8 remainder 11_8.